Optimized load balance scheduling algorithm

Rawaa Mohammed Abdul-Hussein [1], Ahmed Hashim Mohammed [2]
[1]Computer Engineering Department, Faculty of Engineering, Mustansiriyah University, Iraq
[2]Department of Computer Science, College of Education, Mustansiriyah University, Iraq

Article Info

ABSTRACT

The cloud computing environment faces several challenges as a federation of clouds, controlling the traffic flow, scalability, and balancing the load on virtual machines that are considered the most crucial issue due to their impact on the execution time, resource utilization, and cost. This paper is interested in some of the existing algorithms that distribute the workload evenly. These algorithms aim to avoid the blind assignment that often results in some over-loaded servers while another node might be under-loaded. In this work a combination of two inspired metaheuristic algorithms BAT and cuckoo search was proposed; the first algorithm can utilize fast exploration using global search, the latter algorithm can avoid trapping into BAT local optimum problem using levy flight with a far random walk. Additonaly, the proposed algorithm could be used to mitigate distributed denial of service (DDoS) attack that aims to cause endless load on the servers and stop the service. Experimental results for five virtual machine (VM), ten VM, with the varying number of tasks showed that the proposed algorithm has better resource utilization and less makespan time in almost all the cases.

Corresponding Author:

Ahmed H. Mohammed
Department of Computer Science, Mustansiriyah University
Iraq, Baghdad
Email: dr.ahmedh@uomustansiriyah.edu.iq

1. INTRODUCTION

Vast applications and resources storing the data of many users worldwide are provided as a service via a cloud computing system. Although the cloud applications increase rapidly every day, many challenges need to be considered such as security and load balance [1], [2]. The name "cloud" refers to virtual networks that provide software services with high performance at a low cost [3]. Nowadays, cloud computing extends this trend towards hardware, even infrastructure, to be provided as a service [4]. As the requirement of the services was increasing, the need for more parallel and distributed servers will also increase with the need for algorithms to balance the load equally. Without balancing the load there may be some idle nodes while others are fully loaded with a list of demands[5], [6]. The goal of applying these algorithms is to give every server an equal load, so at any time, all the servers are busy with a specified work. This will improve the services by minimizing the response time, improving resource utilization and the overall performance[7]. The main challenge is allocating a suitable resource at the time of a task. These algorithms depend on predicting the number of loads, relations between nodes, implementation of the system. The measured load is different in many aspects, such as central processing unit (CPU) load, the used memory, and time delay[8], [9]. Load balance algorithms can be classified into either static or dynamic resource allocation. In a static algorithm the current state is not considered because it relies on its previous information [10]. Dynamic algorithm depends on present state, and there is no need for prior knowledge of the system. Dynamic algorithm can be classified into distributed and non-distributed algorithm [11]. In distributed algorithm, the work load is harmonized

between all the nodes collectively. It can be applied in two forms: cooperative that relies on each node to reach the goal, and non-cooperative form that balances the load on each node independently[12]. Whereas, non-distributed algorithm, balance the load by one node or node-set. This algorithm is classified into two types: non-distributed centralized that relies, on a centralized node in balancing the load, and semi distributed that divides the node into sets of clusters and inside each group the central node balances the load. These clusters interact to produce the final decision of the whole system [13].

Balancing the load in cloud environment has advantages related to security when hundreds of attackers start to send the unwanted traffic packets in order to acquire the memory, network resources and completely deplete them this attack is known as distributed denial of service (DDoS) attack [14]. One of the most crucial attack is DDoS attacks that are based on a joint attack platform that intends to send incomplete requests traffic that exaust network bandwidth or system resources. resulted in prevention of legitimate users' requests. Addition layer of security can be added by balancing the load as a result of its effect in controlling and avoiding DDoS attacks. Dynamic allocation of resources can be used to mitigate DDoS attack effectivly [15].

Manasrah and Ali [16] have combined the genetic algorithm (GA) and particle swarm optimizations (PSO) optimization algorithm. At first the preliminary solution is found by GA algorithm that is passed to the PSO algorithm to produce the final solution. Results show improvement in the performance (16% over GA and 4% above PSO) by reducing the overall execution time and cost. Fabrizio et al. [17] used asymmetrically clipped optical (ACO) heuristic algorithm to balance heterogeneous load by performing scheduling simultaneously. The tasks graph is mapped and placed on a heterogeneous reconfigurable devices to support PDR based field-programmable gate arrays (FPGAs). The performance comparison shows an improvement of 16.5% over other heuristic algorithms. Jain [18] examined machine learning techniques based on multi-level swam optimization to allocate suitable resources that can utilize continuous data streams, minimize time cost, and increase load balance degree. Hong [19] introduced genetic ant colony algorithm to solve virtual machine (VM) problem by analyzing ant placement between pair of virtual machines (VMs) which is done by monitoring the pheromone during the ant movements. These results will be optimized using a genetic algorithm. The evaluation shows that the physical servers are chosen efficiently resulted in resource utilization improvement. Dave et al. [20] presented PSO for balancing the load running in cloud environment using different applications to generate the load. The comparison shows considerable improvement in the performance of VMs running applications. Awad et al [21] developed load balance mutation particle swarm optimization (LBMPSO) that reassigned failed task and finished scheduling of the distributed tasks as earlier as possible. Results present improvement in round trip time and execution time over other algorithms. Jena [22] proposed multi-objective PSO framework (MOPSO) for scheduling task by combining PSO and an evolutionary algorithm such as a mutation operator with concepts commonly used in multiobjective evolutionary algorithms (MOEAs) based on Pareto dominance with better mechanism for spreading the solutions. The experimental results show improvement in resource utilization and reduction in energy and make span. X. Lu and Z. Gu [23] applied adaptive global expansion factor on ant colony optimization to speed up the convergence process.

The proposed model monitors the nodes to detect the overloaded VM then applying ACO to distribute load on the idle once. The results showed that adaptive cloud resource eliminate hot spots and balance the load efficiently which results in high CPU utilization. Dhinesh and Venkata [24] applied honey bee behavior to balance the load by classifying VMs into over loaded and under loaded nodes then removing the load from the hot spot node (over loaded) node to the idle (under loaded) node according to the priority of each task. This work improves the task execution and waiting time that can be proved by the simulation. Lili and Xu et al. [25] produced a green cloud task algorithm based on binary particle swarm optimization (BPSO). This work uses pipeline number for VMs and reassigns the particle position and velocity instead of using matrix operations. Results showed improvement in the performance by minimizing execution time and resource consumption in VMs. Xue et al. [26] has proposed load balance based on Ant colony optimization by considering the average virtual machine load. The standard ACO pheromone value is updated according to the distance while in the proposed algorithm, the pheromone value is selected according to the computational capabilities. The transformed probability problem of ants is solved by using the roulette algorithm. When tasks have selected the same virtual machine, the algorithm will select other idle virtual machines to minimize the waiting period. Nakrani and Tovey [27] have been inspired by the behavior of some kind of bees known as forager honey bees. This technique is based on the natural procedure that allocates suitable servers to the requested task efficiently. Results showed improvement in the performance over static or greedy algorithm for high request loads, but in low variability greedy algorithm can outperform. Alnusairi et al. [28] has combined particle swarm algorithm with gravitational search algorithm. The proposed algorithm uses PSO exploitation for global search and gravitational search algorithm (GSA)

exploration for local search. The experimental results show that the proposed algorithm balances the load over time and enhances the overall utilization.

The contribution of this paper was in: proposing hybrid algorithm, the first one is BAT algorithm that has the capability of fast convergence and the second algorithm is Cuckoo that overcomes the problem of trapping in local optimum solution. Secondly, this algorithm could be used to mitigate DDoS attack that aims to cause endless load on the servers and stop the service. Balancing the load is the best way to prohibit attackers from DDoS attack by fairly distribution of workloads.

2. PROPOSED HYBRID OPTIMIZATION ALGORITHM

A new hybrid algorithm has been proposed by combining the BAT and Cuckoo metaheuristic Search algorithm. BAT algorithm can reach towards optimum global region quickly (fast exploration) for the best solution over a large population. However, after a sequence of iterations, the local search strategy of BAT algorithm can trap into the local optimum. Consequently, the proposed algorithm uses Cuckoo search algorithm to replace the local search of BAT algorithm by passing the BAT best solution to cuckoo algorithm which uses Lévy flights strategy to overcome BAT local optimum problem.

The proposed algorithm starts to search for a new solution, which is generated in three stages: i) BAT algorithm is applied to find the first solution using global search; ii) Cuckoo algorithm generates a new solution around the best BAT solution; and iii) The proposed algorithm chooses the best solution between BAT and Cuckoo algorithm.

2.1. BAT optimization algorithm

BAT could be classified into three types after studing 1000 species of them: micro BAT, mega BAT, ghost BAT. Micro BAT uses a sonar called echolocation by generating a loud sound wave with low frequency and low pulse rate. When the BAT found the prey, the loudness decreases while the frequency and pulse rate increase with a short time (frequency tuning) to detect the location of prey accurately. This strategy is used in the global BAT search. The BAT algorithm is based on micro BAT behavior, so every BAT is assigned:

− Frequency: number of waves in particular unit time denoted by f_i, the minimum frequency f_{min} and maximum frequency f_{max} of the sound wave will be used to calculate the frequency at each iteration.
− 2-Position: the location of each BAT in the population denoted by x_i.
− Velocity: speed of BAT toward the prey v_i.
− Loudness: the intensity of sound wave A_i, the loudness value range between maximum loudness value A_1 and minimum loudness value A_0, as the BAT becomes closer to the target, the loudness is minimized.
− Pulse rate: the vibration of sound denoted by r_i, as the BAT becomes closer to the prey, the pulse rate is increased.

In the BAT algorithm the searching strategies are classified into two types: i) Local search: is used for generating a new solution around the position of the current best solution, by checking if the random value is greater than the pulse rate; and ii) Global search: a new solution is generated by flying randomly then checking if the fitness of the new solution is lower than the fitness of the current best solution and the loudness value is greater than a random value, then accept the new solution, Increase the pulse rate, and decrease the loudness.

2.2. Cuckoo optimization algorithm

This algorithm is a meta-heuristic algorithm proposed by Yang and Deb in 2009 by analyzing some cuckoo bird behavior that lay their eggs in the nests of other birds. This strange breeding behavior increases their survival and productivity.

Many host birds cannot differentiate Cuckoo bird from their birds, and these host birds will brood cuckoo eggs until they hatch and finally feeding them until chicks grow up. Although many cuckoo birds survive from the discovery of host bird, the possibility that the host bird realizes Cuckoo egg could happen, then host bird may throw away the cuckoo egg from the nest or leave that nest. Three rules should be imposed in the implementation of the Cuckoo algorithm:

− Each cuckoo chooses random nest and puts one egg at a time.
− Best nest represents best solution that is used to predict the next generation of solutions.
− There are a fixed number of nests.

The Best solution represents a high-quality egg (similar to host bird eggs), so it has the opportunity to develop and become a mature Cuckoo. Whereas worse solution represents eggs that could be distinguished by host bird so it should be replaced.

The generation of new solution is done by levy flight which is a set of straight paths turned by 90 degrees each time as the (1):

$$X_i(t+1)=x_i(t) + \alpha \oplus Levy(\lambda) \tag{1}$$

Where i represent cuckoo nest
　　α = step size = 1
　　λ = levy exponent =1.5
　　$\oplus$ = entry wise multiplication
　　$X_i(t+1)$ represent a new solution, $X_i(t)$ represent the current location (solution).

　　　　The evolution process of cuckoo search defines three different stages: i) The first one is Levy (λ) flight that is used to find a new position denoted by $X_i(t+1)$ based on the current location of cuckoo $X_i(t)$ and random step size generated from Levy flight (λ); and ii) The second strategy involves replacing the nest of the discovered egg with another nest. The last strategy is Greedy (elitist) selection by applying the fitness function to extract the best solution. The proposed algorithm can be presented in Figure 1.
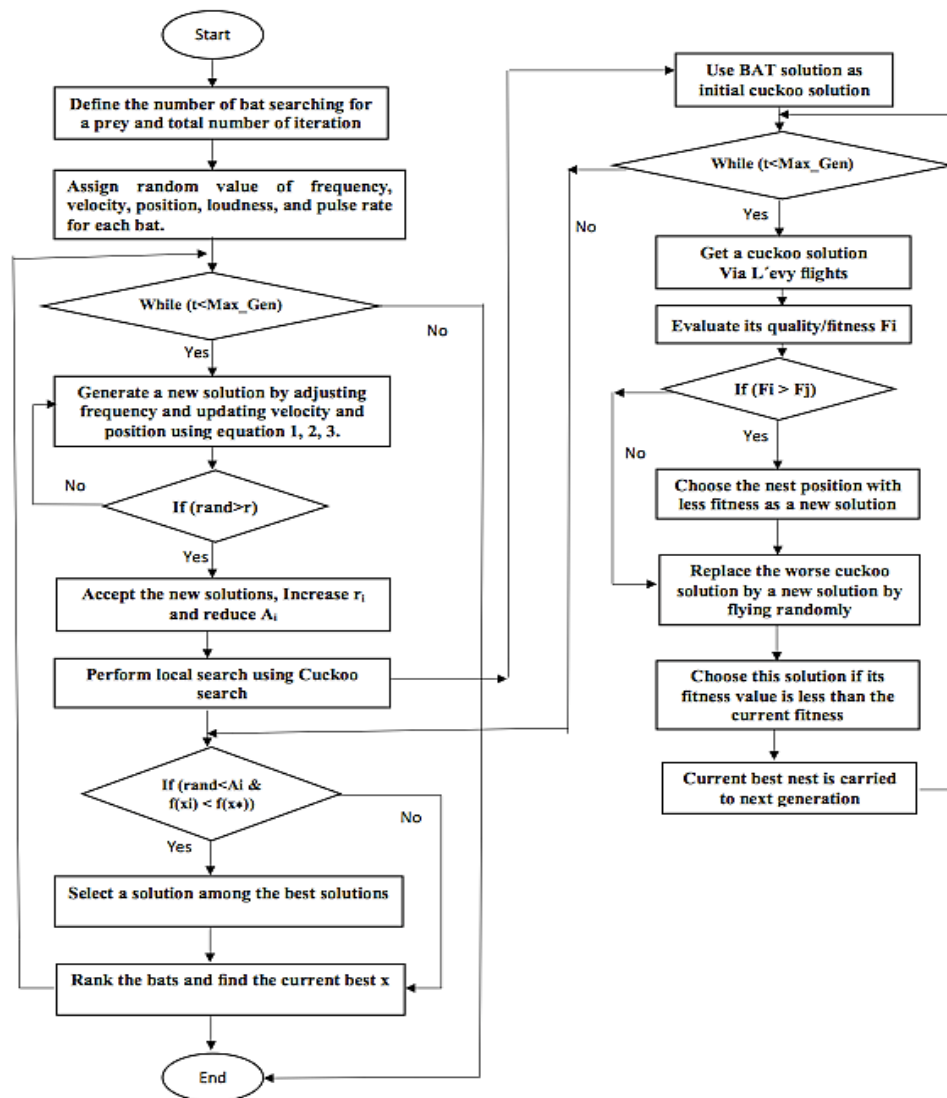
Figure 1. Flowchart of proposed algorithm

　　　　This flowchart can be summarized in the following steps: i) Population: the total number of BATs searching for a prey and total number of iterations; ii) Assign random value of frequency, velocity, position, loudness, and pulse rate for each BAT; and iii) Check, if the numbers of iteration lower than the total number of iterations then generate a new solution by updating the frequency, velocity, and position. Use (1), (2), (3).

$$f_i = f_{min} +(f_{max} - f_{min})\beta \tag{2}$$

$$V_i^t = V_i^{t-1} + (x_i^{t-1} - X^*)f_i \qquad (3)$$

$$x_i^t = x_i^{t-1} + v_i^t \qquad (4)$$

- Check if the random value is greater than the pulse rate then selects this solution among the best solution.
- Apply cuckoo algorithm to generate a new solution around this solution.
- Use the BAT best solution as the initial best Cuckoo solution and calculate its fitness.
- Use L´evy flight to generate new solution and calculate its fitness.
- Compare the new fitness value with current fitness value and choose the nest with less fitness as a new solution.
- Replace the worse cuckoo solution by a new solution by flying randomly, choose this solution if its fitness value is less than the current fitness.
- return to BAT algorithm and Check if the fitness of the new cuckoo solution is lower than the fitness of BAT best solution and the loudness value is greater than a random value then:
- Decrease the loudness and increase the pulse rate according to the following equations

$$A_i^{(t+1)} = \alpha A_i^{(t)} \qquad (5)$$

$$r_j^{(t)} = r_i^{(0)}[1 - exp(-y\epsilon)] \qquad (6)$$

ϵ is a random number from $[-1, 1]$, $A_i^{(t)}$ is the average loudness of the population.
- Rank the BAT and accept the best new solution with higher frequency and pulse rate.
- Repeat the above steps until termination criteria satisfied.
As a result, the proposed algorithm can explore wide range of problem space while avoiding getting stuck in local optimum.

3. RESULTS AND DISCUSSIONS

Cloud sim has been used to investigate large-scale cloud environment.it is developed by the Grid bus project team of Melbourne University in Australia. The implementation of the proposed algorithm is done by using Java programming language, IDE: Eclipse. The evaluation is measured by comparing a set of parameters like execution time and average utilization over 5 and 10 VMs for varying number of cloudlets (tasks). The proposed algorithm is compared with Round Robin Algorithm (RR), first come first served (FCFS) and standard BAT algorithm. Round Robin assign task to each virtual machine in Sequential order, so when the first task arrive, it will be sent to the first virtual machine then the second task is passed to the second virtual machine and so on. FCFS algorithm allocates the load in a sequential order according to its precedence of arrival. The evaluation is based on comparing Make Span, which is known as job completion time. It is measured in nanoseconds. The experimental result can be categorized into five cases:

3.1. Execution time for VMs

As shown in Figure 2, there is small difference between the execution time when the range of task is between 10 to 20. When the number of tasks exceeds 30 tasks, the differences in execution time increases gradually. The improvement of the proposed BAT cuckoo search (BATCS) has reached to about 14 % over FCFS algorithm, 6% over RR, and 4% over standard BAT algorithm.

For 10 VMs, the execution time is presented in Figure 3. It can be observed from Figure 2 that the proposed BATCS algorithm has minimum execution time over all other algorithms. The proposed BATCS achieves about 18 % less execution time than RR algorithm and 9% over both FCFS and standard BAT algorithm.

3.2. CPU utilization for VMs

As presented in Figure 4, when applying five virtual machine to the simulation, the results prove that BATCS algorithm improve the average resource utilization when compared to FCFS, RR, and standard BAT algorithms by 8%, 8%, and 4%, respectively. The CPU utilization results obtained for 10 VMs are shown in Figure 5. The average improvement of this comparison indicates that BATCS algorithm has higher utilization than FCFS, RR with 13 % and around 6 % over standard BAT algorithms.

3.3. Degree of Imbalance

Figure 6. presents the degree of imbalance for MakeSpan time. It can be observed that after balancing the load, the makespan time increases smoothly and gradually from about (2 to 7) nanosecond over a range of (10 to 40) tasks. Whereas before balancing the load, the variation in time shows unstable increase,

beginning from just over 5 nanosecond at 10 task, to well under 30 nanosecond at 40 task. It can be concluded that the improvement in the degree of imbalance before and after balancing the load reaches to 60 %. Similarly the average execution time for varying number of task over 5 and 10 VMs can be consolidated in Figure 7.
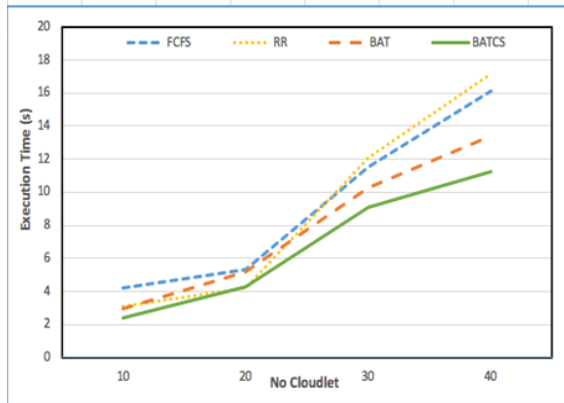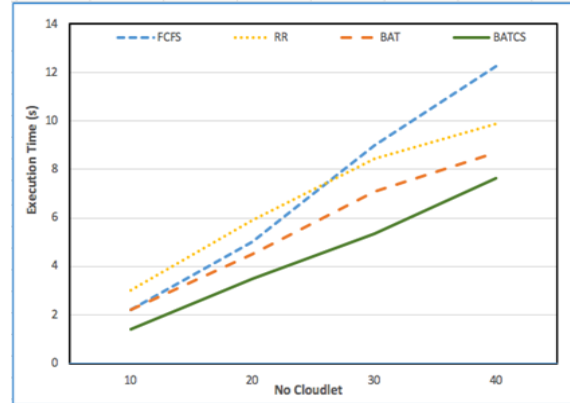

Figure 2. Execution time of (5 VMs) VS Tasks

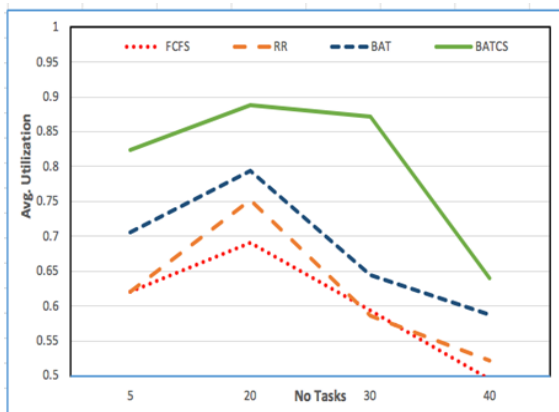Figure 3. Execution time of (10 VMs) VS Tasks
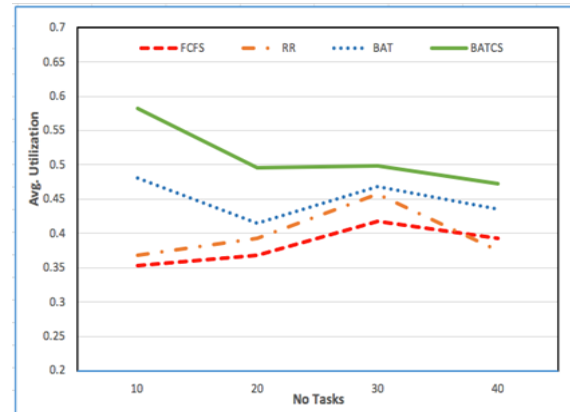

Figure 4. Avg Utitization of (5 VMs) VS Tasks

Figure 5. Avg Utilization of (10 VMs) VS Tasks
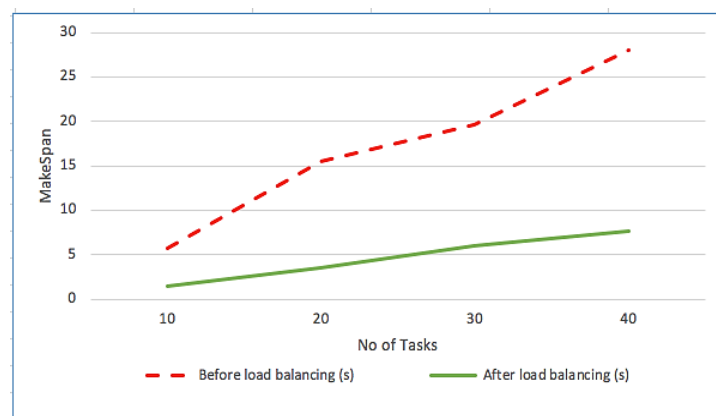
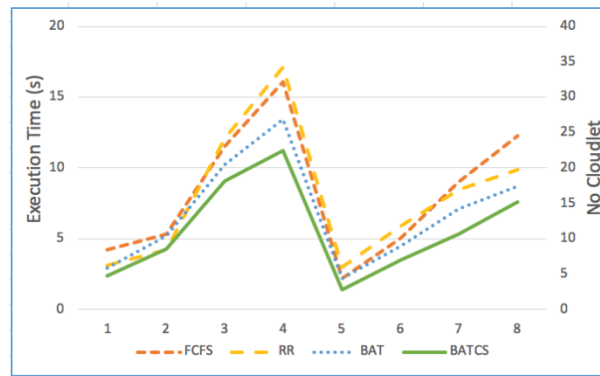Figure 6. Degree of inbalance before load balance

Figure 7. Exceution time of (5-10) VMs VS Tasks

4.　CONCLUSION

In cloud environment the load should be balanced efficiently by appling smart algorithms. Today scientists relay on metaheuristic algorithm to improve the performance by selecting the suitable algorithms skillfully. This work uses BAT algorithm that have high capability of global convergence to solution in short time, then after several iterations the new solutions may fall into the local optimum problem. Therefore, this work suggests applying cuckoo algorithm as a second stage that can find best new solution far from the current best solution. Results showed improvement in execution time by about 14 % over FCFS algorithm, 6% over RR, and 4% over standard BAT algorithm for 5 VMs and 18 % less execution time than RR algorithm and 9% over both FCFS and standard BAT algorithm for 10 VMs. Moreover, the proposed algorithm achieves higher resource utilization for 5 VMs when compared to FCFS, RR, standard BAT algorithms by 8%, 8%, and 4%, respectively. Additionally, BATCS algorithm has higher utilization than FCFS, RR with 13 % and around 6 % over standard BAT algorithms. Finally, the performance comparison showed that the improvement in the degree of imbalance before and after balancing the load has reached to 60%.

On the other hand, the proposed algorithm could be used to mitigate DDoS attack that aims to cause endless load on the servers and stop the service. Balancing the load is the best way to prohibit attackers from DDoS attack by distributing the workload fairly. As a future work the data generated in case of normal state (without attack) can be used to make a comparision by appling machine learning algorithms to distinguish between legitimate and aggressive legitimate users.

REFERENCES

[1]　M. Gamal, R. Rizk, H. Mahdi, and B. E. Elnaghi, "Osmotic Bio-Inspired Load Balancing Algorithm in Cloud Computing," *IEEE Access*, vol. 7, pp. 42735–42744, 2019, doi :10.1109/ACCESS.2019.2907615.
[2]　I. Ahmed, "Technology organization environment framework in cloud computing," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, n0. 2, pp. 716–725, doi: http://dx.doi.org/10.12928/telkomnika.v18i2.13871.
[3]　Y. Sahu and R. K. Pateriya, "Cloud Computing Overview with Load Balancing Techniques*," International Journal of Computer Applications*, vol. 65, no. 24, 2013, [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.302.9773&rep=rep1&type=pdf
[4]　Alyouzbaki, Y. A. G. and Al-Rawi, M. F. "Novel load balancing approach based on ant colony optimization technique in cloud computing*," Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp. 2320–2326, 2021, doi: https://doi.org/10.11591/eei.v10i4.2947.
[5]　D. A. Shafiq, N. Z. Jhanjhi, and A. Abdullah, "Load balancing techniques in cloud computing environment: A review," *Journal of King Saud University- Computer Information Sciences*, 2021, doi: https://doi.org/10.1016/j.jksuci.2021.02.007.
[6]　T. Francis, "A Comparison of Cloud Execution Mechanisms Fog, Edge, and Clone Cloud Computing," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no.6, pp. 4646-4653,2018, doi: http://doi.org/10.11591/ijece.v8i6.pp4646-4653.
[7]　N. Haryani and D. Jagli, "Dynamic Method for Load Balancing in Cloud Computing," IOSR Journal of Computer Engineering (IOSR-JCE), vol. 16, no. 4, pp. 23-28, 2014, [Online]. Available: https://www.iosrjournals.org/iosr-jce/papers/Vol16-issue4/Version-4/D016442328.pdf
[8]　A. Kaur and M. P. Luthra, "A Review on Load Balancing In Cloud Environment", *International Journal Of Computers & Technology (IJCT)*, vol. 17, no. 1, pp. 7120–7125, 2018, doi: https://doi.org/10.24297/ijct.v17i1.7160.
[9]　C. Jittawiriyanukoon, "Cloud computing based load balancing algorithm for erlang concurrent traffic," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 17, no.6, pp. 1109–1116, 2019, doi: http://doi.org/10.11591/ijeecs.v17.i2.pp1109-1116.
[10]　R. Pushpa and M. Siddappa, "A comparative study on load-balancing algorithms/or cloud environments," 2017 3rd *International*

Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2017, pp. 316-321, doi: 10.1109/ICATCCT.2017.8389154.

[11] R.Kaur and G. Kaur, "Proactive scheduling in cloud computing," *Bulletin of Electrical Engineering and Informatics*, vol. 6 no. 2, pp. 174–180, 2017, doi: 10.11591/eei.v6i2.649.

[12] I. Odun-Ayo, T. -A. Williams, M. Odusami, and J. Yahaya, "A systematic mapping study of performance analysis and modelling of cloud systems and applications," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 2, pp. 1839–1848, 2021, doi: 10.11591/ijece.v11i2.pp1839-1848.

[13] A.Ullah, N. M. Nawi, J. Uddin, S. Baseer, and A. H. Rashed, "Artificial bee colony algorithm used for load balancing in cloud computing: Review," *International Journal of Artificial Intelligence* ,vol. 8, no. 2, pp. 156–167, 2019, doi: 10.11591/ijai.v8.i2.pp156-167.

[14] N. Agrawal and S. Tapaswi, "A proactive defense method for the stealthy EDoS attacks in a cloud environment," *International Journal of Network Management*, vol. 30, no. 2, pp. 1–25, 2020, doi: 10.1002/nem.2094.

[15] N. Jeyanthi, N. Ch. S. N. Iyengar, P. C. M. Kumar, and Kannamal A, "An enhanced entropy approach to detect and prevent DDOS in cloud environment," *International Journal of Communication Networks and Information Security*, vol. 5, no. 2, pp. 110–119, 2013, doi: 10.54039/ijcnis.v5i2.367.

[16] A. M. Manasrah and H. B. Ali, "Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing", *Wireless Communications and Mobile Computing*, 2018, doi: 10.1155/2018/1934784.

[17] F. Ferrandi, P. L. Lanzi, C. Pilato, D. Sciuto, and A. Tumeo, "Ant Colony Optimization for mapping, scheduling and placing in reconfigurable systems," *2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)*, 2013, pp. 47-54, doi: 10.1109/AHS.2013.6604225.

[18] A. Jain, "Advance Approach for Load Balancing in Cloud Computing Using (HMSO) Hybrid Multi Swarm Optimization," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 10, pp. 46-49, 2018.

[19] L. Hong and G. Yufei, "GACA-VMP: Virtual Machine Placement Scheduling in Cloud Computing Based on Genetic Ant Colony Algorithm Approach," *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, 2015, pp. 1008-1015, doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.189.

[20] A. Dave, B. Patel, G. Bhatt, and Y. Vora, "Load balancing in cloud computing using particle swarm optimization on Xen Server," *2017 Nirma University International Conference on Engineering (NUiCONE)*, 2017, pp. 1-6, doi: 10.1109/NUICONE.2017.8325618.

[21] A. I. Awad, N. A. El-Hefnawy, and H. M. A. Kader , "Enhanced Particle Swarm Optimization for Task Scheduling in Cloud Computing Environments," *Procedia Computer Science*, vol. 65, pp. 920–929, 2015, doi: 10.1016/j.procs.2015.09.064.

[22] R. K. Jena, "Multi Objective Task Scheduling in Cloud Environment Using Nested PSO Framework," *Procedia Computer Science*, vol. 57, pp. 1219–1227, 2015, doi: 10.1016/j.procs.2015.07.419.

[23] X. Lu and Z. Gu, "A load-adaptive cloud resource scheduling model based on ant colony algorithm," *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, 2011, pp. 296-300, doi: 10.1109/CCIS.2011.6045078.

[24] L. D. Dhinesh and P. Venkata, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Applied Soft Computing Journal, vol. 13, no. 5, pp. 2292–2303, 2013, doi: 10.1016/j.asoc.2013.01.025.

[25] L. Xu, K. Wang, Z. Ouyang, and X. Qi, "An improved binary PSO-based task scheduling algorithm in green cloud computing," *9th International Conference on Communications and Networking in China*, 2014, pp. 126-131, doi: 10.1109/CHINACOM.2014.7054272.

[26] S. Xue, M. Li, X. Xu, and J. Chen, "An ACO-LB algorithm for task scheduling in the cloud environment," J. Softw., vol. 9, no. 2, pp. 466–473, 2014, doi: 10.4304/jsw.9.2.466-473.

[27] S. Nakrani and C. Tovey, "On Honey Bees and Dynamic Server Allocation in Internet Hosting Centers," *Adaptive behavior*, vol. 12, pp. 223-240, 2004, doi:10.1177/105971230401200308.

[28] T. S. Alnusairi, A. A. Shahin, and Y. Daadaa, "Binary PSOGSA for Load Balancing Task Scheduling in Cloud Environment," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 5, pp. 255- 264, 2018, doi: 10.14569/IJACSA.2018.090535.

BIOGRAPHIES OF AUTHORS

Rawaa Mohammed Abdul-Hussein ⓘ 🔳 SC Ⓟ received the Bachelor degree of "computer and software engineering" from university of Mustansiriyah. From 2006-2008, she received a Master degree of "information technology" from University of Technology. Her Research interests are cloud computing, software engineering, intelligent algorithm, logic design, and web application security. She can be contacted at email: mscrawaahm@uomustansiriyah.edu.iq.

Ahmed Hashim Mohammed ⓘ 🔳 SC Ⓟ received computer Science degree from Al-Rafidain University, Baghdad, Iraq, in 2003 and M.Sc. degree in computer science from Informatics Institute for Postgraduate Studies, Baghdad, Iraq, in 2006, and Ph. D degree in computer science from University of Technology, Baghdad, Iraq in 2015. He is Currently Assist Prof. lecturer at Al- Mustansiriyah University. His research interests include Artificial intelligent, Cloud Computing, cyber security, internet of things and cryptanalysis. He can be contacted at email: dr.ahmedh@uomustansiriyah.edu.iq.